PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-224951
(43)Date of publication of application : 03.09.1993

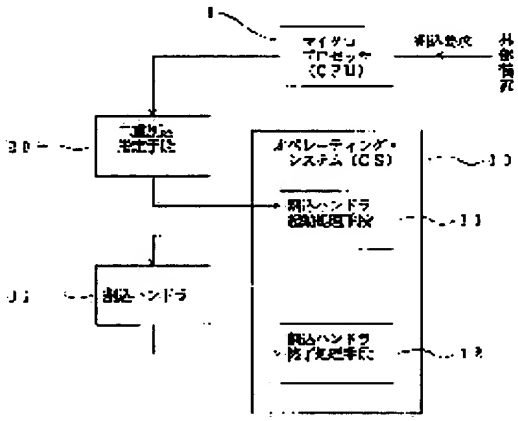
(51)Int.CI. G06F 9/46
G06F 13/24

(21)Application number : 04-024226
(22)Date of filing : 12.02.1992

(71)Applicant : FUJITSU COMMUN SYST LTD
(72)Inventor : NAKAJIMA YASUO
YAMAWAKI TOMOHIRO
ATAMI NOBUO

(54) INTERRUPTION MANAGING METHOD FOR MICROPROCESSOR SYSTEM

(57)Abstract:
PURPOSE: To provide the interruption managing method for generating easily an interruption handler, and decreasing an overhead of an OS.
CONSTITUTION: In the microprocessor system for suspending a processing at the time when a CPU 1 which is executing a processing through an OS 10, receives an interruption request from an external device, and allowing an interruption handler 31 to execute an interruption processing, this system is constituted by providing a double interruption designating means 20 for designating a soft interruption at the time when it is actuated from the CPU which receives the interruption request and migrating the processing to the OS before migrating the processing to the interruption handler, an interruption handler actuation processing means 11 for analyzing an interruption factor and retaining a register, etc., in the OS and actuating the interruption handler, and an interruption handler finish-processing means 12 for executing reset of the register, etc., and an interruption finish processing to the external device in the OS at the time when an end of the interruption processing is informed from the interruption handler, and thereafter, restarting the suspended processing.



LEGAL STATUS

[Date of request for examination] 06.01.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平5－224951

(43)公開日　平成５年(1993)９月３日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/46	３１０　Ｚ	8120－5B		
	13/24	３１０　Ｚ	9072－5B		

審査請求　未請求　請求項の数１(全 14 頁)

(21)出願番号　特願平4－24226

(22)出願日　平成４年(1992)２月12日

(71)出願人　000237651
富士通コミュニケーション・システムズ株式会社
神奈川県横浜市港北区新横浜３丁目９番18号

(72)発明者　中島　康夫
神奈川県横浜市港北区新横浜三丁目９番18号　富士通コミュニケーション・システムズ株式会社内

(74)代理人　弁理士　井桁　貞一

最終頁に続く

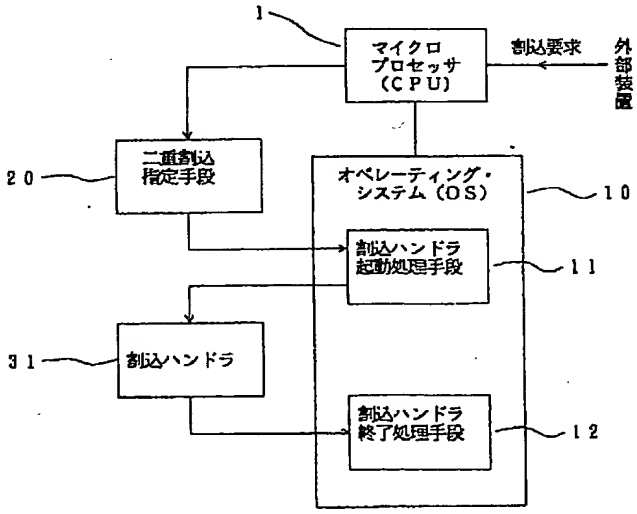
(54)【発明の名称】　マイクロプロセッサ・システムの割込管理方法

(57)【要約】

【目的】　マイクロプロセッサ・システムの割込管理方法に関し、割込ハンドラの作成が容易で、ＯＳのオーバーヘッドが少ない割込管理方法を提供することを目的とする。

【構成】　ＯＳ10を介して処理実行中のＣＰＵ１が外部装置より割込要求を受けたときに処理を中断して割込ハンドラ31に割込処理を行わさせるマイクロプロセッサ・システムにおいて、割込要求を受けたＣＰＵより起動されたときにソフト割込を指定して割込ハンドラへの処理移行前にＯＳに処理を移行させる二重割込指定手段20と、二重割込指定手段を介して処理を移行されたとき、ＯＳ内で割込要因の解析とレジスタ類の保存を行って割込ハンドラを起動する割込ハンドラ起動処理手段11と、割込ハンドラより割込処理の終了を通知されたときにＯＳ内でレジスタ類の復帰と外部装置に対する割込終了処理を行ったのちに中断した処理を再開させる割込ハンドラ終了処理手段12を備えるように構成する。

本発明の基本構成図



【特許請求の範囲】
【請求項１】　オペレーティングシステム(10)を介してプログラム処理を実行中のマイクロプロセッサ(1) が外部装置より割込要求を受けたときに実行中のプログラム処理を中断して割込要因ごとの処理を規定する割込ハンドラ(31)に割込処理を行わさせるマイクロプロセッサ・システムにおいて、
割込要求を受けた前記マイクロプロセッサ(1) より起動されたときにソフト割込を指定して前記割込ハンドラ(31)への処理移行前に前記オペレーティングシステム(10)に処理を移行させる二重割込指定手段(20)と、
前記オペレーティングシステム(10)内において、前記二重割込指定手段(20)を介して処理を移行されたときに、割込要因の解析と前記マイクロプロセッサ(1)の内部レジスタ類の保存を行ったのち、前記割込ハンドラ(31)を起動する割込ハンドラ起動処理手段(11)と、
前記オペレーティングシステム(10)内において、前記割込ハンドラ(31)より割込処理の終了を通知されたときに、前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる割込ハンドラ終了処理手段(12)を備えたことを特徴とするマイクロプロセッサ・システムの割込管理方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はマイクロプロセッサ・システムの割込管理方法に関する。近年、多くのマイクロプロセッサ・システムが使用されているが、マイクロプロセッサ・システムにおいてはマイクロプロセッサ（以下、ＣＰＵと記す）がプログラムの処理を実行中に外部デバイスまたは外部装置（以下、外部装置と総称する）から割込要求が行われたときに実行中のプログラム処理を中断して外部装置の割込要求を処理することが多い。
【０００２】外部要求で処理する内容は外部装置ごとに異なるので、ユーザが外部装置ごとに割込処理ルーチン（以下、割込ハンドラと記す）を作成しておき、割込要求があったときにＣＰＵが割込要求を行った外部装置を識別し、該当する割込ハンドラを呼び出して処理させる方法が用いられている。
【０００３】外部割込の発生によりＣＰＵが実行中のプログラム処理を中断する際には、再開するプログラムの番地や、それまで使用していたＣＰＵの内部レジスタの内容を保存してから割込処理を実行し、割込処理が終了したときにこれらのレジスタを元の状態に復してプログラムを再開するが、前記割込ハンドラの中で内部レジスタの退避処理を記述することは割込ハンドラの作成を煩雑にするため、これらの処理はＯＳ（オペレーティング・システム）の割込管理機能によって行うのが普通である。
【０００４】しかし、現在普及しているマイクロプロセッサ・システムでは割込要求を受けたＣＰＵがハード的に割込ハンドラを起動するようになっているため、もしそのまま割込ハンドラによって割込処理が開始されるとＯＳは割り込みの発生を知ることができず、レジスタ類の保存を行うこともできなくなる。このため、割込ハンドラ側からＯＳを呼び出して（システムコールと呼ぶ）ＯＳにレジスタの保存などの処理を行ってもらい、これらの処理の終了後に再び割込ハンドラに処理を戻して割込処理を行う方法が用いられている。割込処理の終了時も同様に割込ハンドラからシステムコールを行い、ＯＳがレジスタ等の復帰を行ったのち、中断していたプログラムを再開させる。なお、このシステムコールによってＯＳは割込処理の状態を管理することが可能となっている。
【０００５】一方、ユーザが作成する割込ハンドラは現在では高級言語によって書かれるのが普通であるが、高級言語によって前記のシステムコールを含めて記述すると、機械語に翻訳した時点で、モジュール（ハンドラ）の先頭で必ずスタックの操作と、いくつかのＣＰＵ内レジスタの書き替えが行われるようになっている。このため、ＣＰＵのレジスタは内容を退避する前に使用されることとなる。割込処理の終了時にも同様な問題が発生する。
【０００６】このような状態を避けるためにはシステムコールに前後する部分を高級言語でなくアセンブラ言語を用いて記述する必要がある。しかし、現在ではアセンブラ言語による記述は一般的でないため、正確な記述を行うのが難しく、加えて割込ハンドラの内部で分岐などがあると終了時の処理に誤りが生じ易い。
【０００７】このため、割込ハンドラを高級言語のみで記述することが可能で、処理誤りが発生し難いマイクロプロセッサ・システムの割込管理方法が必要となっている。
【０００８】
【従来の技術】図６はマイクロプロセッサ・システム構成図、図７は従来技術のシステム構成図、図８は従来技術のメモリ構成図、図９は従来技術の処理移行状態の説明図、図10は従来技術のフロー図である。
【０００９】図６は本発明が適用されるマイクロプロセッサ・システムの構成の一例を示している。図６のＣＰＵ１は８０８６シリーズのマイクロプロセッサ（ＣＰＵ）を用いて構成されたマイクロプロセッサ・システムの例を示しているが、ＣＰＵ１はバス６を介してＲＡＭ（ランダムアクセス・メモリ）４、ＲＯＭ（読出専用メモリ）５、割込コントローラ（以下、ＰＩＣと記す）２及び８台の外部装置（Ｉ／Ｏ）3_{-1}～3_{-8}と接続されている。
【００１０】ＰＩＣ２は８台の外部装置3_{-1}～3_{-8}のうちの複数からIR_0 ～IR_7 線を介して同時に割込要求があった場合に優先順位などを比較し、同時に一つの割

込要求のみがＣＰＵ１に行われるように調停する役割をもち、割込要求はＩＮＴＲ線を通してＣＰＵ１に送出される。１つのＰＩＣ２には８台の外部装置が接続できるが、ＰＩＣ２をカスケードに接続することにより最大２５６台までの外部装置を接続することができる。

【００１１】図７は従来技術におけるマイクロプロセッサ・システムの割込処理関連部分のシステム構成を図示し、図８は図７の各部のメモリ（図７では図示省略）上の記憶状態の例を図示しているが、以下、図７及び図８により従来技術を説明する。なお、図７及び図８では２台の外部装置3_{-1}～3_{-2}のみを記載している。

【００１２】図７の構成部分の一部は図８に示すようにＲＡＭとＲＯＭにより構成されるメモリ上に記憶され、ＣＰＵ１がこれらを読み出しながら各機能を動作させる。プログラム類はＲＡＭまたはＲＯＭのいずれにも記憶可能なものが多いが、ここでは説明の便から、ＲＯＭにはＯＳ60を構成する割込ハンドラ起動処理部61、割込ハンドラ終了処理部62及び実行管理部63のみが記憶され、割込ハンドラ81及びアプリケーション・プログラム（以下、ＡＰと記す）82はＲＡＭに記憶されるものとする。なお、以下において単にＯＳと記した場合にはＯＳ60全体またはその一部（主として実行管理部63）を指すものとする。

【００１３】いま、ＣＰＵ１がＯＳの実行管理部63との共同動作によりＡＰ82の1200番地の命令を実行中であるとすると、ＣＰＵ１内のプログラムカウンタ（以下、ＰＣと記す）1aは図８に示すように実行中の番地の次の1201番地を指している。

【００１４】この状態で外部装置3_{-1}より割込要求があり、ＰＩＣ２が他の割込要求と優先度などを比較してこの割込要求を優先することを確認すると、これをＣＰＵ１に伝える。割込要求の内容は外部装置ごとに異なるため、どの外部装置からの割込要求であるかを識別する必要があるが、以下、割込要求を行った外部装置を割込番号で識別することとし、外部装置3_{-1}及び3_{-2}の割込要求をそれぞれ割込番号１及び割込番号２とする。

【００１５】図８の割込ベクタ領域71は各割込番号の割込処理内容を規定する割込ハンドラ81が記憶されているメモリ領域の先頭番地（ベクタと呼ばれる）を記憶しているメモリ領域であり、割込番号ごとに４バイト（セグメントとオフセット各２バイト）のアドレス情報を記憶している。

【００１６】外部装置は最大２５６まで接続できるため、割込ベクタ領域71には０から２５５までの番号をもつ２５６の領域が確保されている。この番号をタイプナンバと呼び、各タイプナンバへの割り込みをタイプ０割込～タイプ２５５割込と呼んでいる。実際にはユーザが使用できるタイプナンバの範囲は限定されているが、ここでは説明の便から、割込番号１をタイプ０割込、割込番号２をタイプ１割込に対応させて説明する。

【００１７】ベクタ領域71の物理アドレス（バイトアドレス）には例えば 00000～ 003FF番地（１６進数）が使用され、各タイプナンバごとに割り当てられた４バイトの先頭番地がアドレスとして使用される。従って、割込番号１は 00000番地、割込番号２は 00004番地を使用して割込ベクタ領域71にアクセスする。

【００１８】外部装置3_{-1}の割込要求がＰＩＣ２を介してＣＰＵ１に伝えられると、ＣＰＵ１は図６のＩＮＴＡ線に割込要求を受け付けたことを知らせ、ＰＩＣ２より割込要求元を識別する情報として前記タイプ０割込のアドレス情報 "00000"が送られる。これによりＣＰＵ１は00000番地にアクセスし、その内容を読み取る。

【００１９】タイプ０割込に記憶されている割込ハンドラの先頭番地が 3000番地であるとすると、ＣＰＵ１はこの番地、即ち、割込ハンドラ81の処理に移るが、それに先立ち、中断するＡＰ82の情報を退避させる。

【００２０】図８では退避先のメモリ領域はＲＡＭ上のスタック領域83に確保されており、ＣＰＵ1 内のスタックポインタ（以下、ＳＰと記す）1bが退避情報を記憶させる番地を指している。退避する情報は割込処理が終了したときに復帰すべき番地であり、この場合はＰＣ1aが指しているアドレスの"1201"となる。

【００２１】ＣＰＵ１がこの"1201"をスタック領域83に記憶させるとＳＰ1bは次の番地に移動し、更に高位の割り込みに備える。これを終了したのち、ＰＣ1aは割込番号１用の割込ハンドラの先頭番地である3000番地に跳び、処理が割込ハンドラ81に移る。ここまでの処理はＯＳ60が関与せずにＣＰＵ１によって行われる。

【００２２】割込ハンドラ81は処理を開始すると、先ず前述したシステムコールを行うのでＣＰＵ１はＯＳ60の一部を構成する割込ハンドラ起動処理部61によりレジスタ類（図示省略）の退避、即ち、ＡＰ82で使用していたレジスタの内容の退避を開始する。

【００２３】このとき、ＯＳ60は割込ハンドラ81から起動されたが、起動した割込ハンドラがどの外部装置のものであるか知らされていないため、ＰＩＣ２にアクセスし、その内部レジスタ（図示省略）を読むことによって確認する。これによってＯＳは、現在どの割込処理が行われているかを管理することが可能となり、多重割込が行われた場合などに対処することができる。しかし、この割込要因の解析は、外部装置数が多く、ＰＩＣ２がカスケード接続されているような場合には時間を要し、ＯＳのオーバーヘッドが大きくなる原因となる。

【００２４】内部レジスタ類の保存と割込要因の解析を終了すると、ＣＰＵ１は再び処理の実行を割込ハンドラ81に移し、割込ハンドラ81によって実質的な割込処理が行われる。なお、割込ハンドラ81による割込処理の実行中に高位の割込が発生すると、ＣＰＵ１及びＯＳは前記と同様な処理により高位の割込ハンドラを起動し、その割込処理を実行させるたのち、前の割込ハンドラに処理

を戻す。ＯＳはこのような多重割込などに備え、常に割込状態を管理する。

【００２５】割込処理が終了すると割込ハンドラ81は再びシステムコールをかけ、これによりＯＳの割込ハンドラ終了処理部62がレジスタ類の復帰やＰＩＣ２の終了処理を行い、スタック領域83に記憶されているアドレス情報"1201"をＰＣ1aに移すことにより中断していたＡＰ82の処理が再開される。

【００２６】以上においては説明を省略したが、現在のマイクロプロセッサ・システムが使用できるメモリ空間は極めて大きいため、アドレスの桁数が長大になることを避ける手段としてアドレスをセグメント（他と区別する場合はデータ・セグメントと呼ばれる）とオフセットに分け、セグメントを予めタスクごとに割り当てておく方法が使用されている。これにより、各プログラムはセグメントを意識することなく、オフセットのみを使用して作成することができる。しかし、アプリケーション・プログラムの処理から割込処理に移る場合にはセグメントが変わることとなるため、割り込みの際には割込ハンドラ81とＯＳ60の間でセグメントの付け替えについての手順が取り決められる。

【００２７】図９は割込処理の際の処理の移行状態を示しているが、図示のように、ＡＰの処理実行中に割込要求が発生すると、ＡＰの処理が中断されて割込ハンドラが起動される。割込ハンドラは起動されるとシステムコールによってＯＳを呼び出し、ＯＳによるレジスタの保存などが終了するとＯＳより実行が移されて本来の割込処理を実行する。割込処理を終了すると再びシステムコールを行い、以後ＯＳがレジスタの復帰などを行って中断していたＡＰの処理を再開する。

【００２８】図10は以上におけるＣＰＵとＯＳの処理のフロー図で、Ｓ11～Ｓ14及びＳ21～Ｓ23はフローの各ステップを示す記号である。図10の(1) は割込ハンドラ起動時のフローを示しているが、割込処理の開始時は図示のように、割込ベクタにより割込ハンドラを起動し（Ｓ11）、割込ハンドラよりのシステムコールによりＯＳの割込ハンドラ起動処理部がＣＰＵ内部レジスタの保存（Ｓ12）と割込要因の解析（Ｓ13）を行い、処理を割込ハンドラに移して（Ｓ14）処理を終了する。

【００２９】図10の(2) は割込ハンドラ終了時のフローであり、割込ハンドラよりのシステムコールにより処理の終了を通知された割込ハンドラ終了処理部が終了処理を行うが、多重割込が行われていた場合はスタック領域に保存されていた中断中の割込ハンドラのアドレスを管理領域に復帰させ（Ｓ21）、ＰＩＣ２の終了処理（Ｓ22）と内部レジスタの復帰（Ｓ23）を行い、中断していた処理を再開する。

【００３０】

【発明が解決しようとする課題】以上のように、従来技術ではＯＳがもつ割込要因解析機能、レジスタ保存機能及び割込状態管理機能を用いてユーザが作成した割込ハンドラに割込処理を行わせているが、次のような問題を有している。

【００３１】(1)　ユーザが作成する割込ハンドラは起動後にシステムコールを行ってＯＳにレジスタの保存などを行わせているが、割込ハンドラを高級言語で記述するとシステムコールの際に先ず処理に必要な数値を転送し、保存し終わっていないレジスタを使用する可能性があるため、高級言語のみで割込ハンドラを作成することができない。

【００３２】(2)　割込ハンドラの中で分岐などが行われていると、終了処理が正確に行われず、終了時のシステムコールに誤りが発生し易いため、ユーザは割り込みの状態を意識して割込ハンドラを作成する必要がある。

【００３３】(3)　割り込みの際にデータ・セグメントを変更する必要が生ずるため、ＯＳとの間でデータ・セグメントの設定についての手順が必要となり、割込ハンドラの作成が複雑となる。

【００３４】(4)　割込要因の解析の際にＯＳのオーバーヘッドが大きくなる。即ち、従来技術は割込ハンドラの作成が難しく、ＯＳのオーバーヘッドが大きいという欠点を有している。

【００３５】本発明は、割込ハンドラの作成が容易で、ＯＳのオーバーヘッドが少ない割込管理方法を提供することを目的とする。

【００３６】

【課題を解決するための手段】図１は本発明の基本構成図である。図中、１はマイクロプロセッサ（ＣＰＵ）、10はオペレーティング・システム（ＯＳ）、31は外部割込における割込要因ごとの処理を規定する割込ハンドラ、20は外部装置（図示省略）より割込要求を受けた前記マイクロプロセッサ１より起動されたときにソフト割込を指定して前記割込ハンドラ31への処理移行前に前記オペレーティングシステム10に処理を移行させる二重割込指定手段である。

【００３７】11は前記オペレーティングシステム10内において、前記二重割込指定手段20を介して処理を移行されたときに、割込要因の解析と前記マイクロプロセッサ１の内部レジスタ類（図示省略）の保存を行ったのち、前記割込ハンドラ31を起動する割込ハンドラ起動処理手段、12は前記オペレーティングシステム10内において、前記割込ハンドラ31より割込処理の終了を通知されたときに、前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる割込ハンドラ終了処理手段である。

【００３８】

【作用】図１において、ＯＳ10を介してプログラム処理を実行中のＣＰＵ１が外部装置より割込要求を受けると、実行中のプログラム処理を中断して二重割込指定手段20にアクセスする。

【００３９】二重割込指定手段20はＣＰＵ１に対して割込ハンドラ31が記憶されている番地を直接知らせずに、ソフト割込の実行を指定する。このため、外部装置よりの割込要求により直接割込ハンドラ31に処理が移されず、ソフト割込処理が開始される。

【００４０】ソフト割込処理は外部割込に優先して行われるが、その際、ユーザが作成した割込ハンドラ31の代わりにＯＳ10内部において割込処理の役割をもつ割込ハンドラ起動処理手段11が記憶されている番地を指定する。これにより、割込要求が行われたときに、ＣＰＵ１のハード的な動作によって割込ハンドラ31に直接処理が移されずにＯＳ10に処理が移される。

【００４１】処理を移されたＯＳ10内の割込ハンドラ起動処理手段11は割込要因の解析とＣＰＵ１の内部レジスタ類の保存を行ったのち、前記割込ハンドラ31を起動する。前記割込要因の解析によって割込要求を行った外部装置が識別され、以後、ＯＳが割込状態を管理することが可能となるが、この割込要因の解析は外部装置側にアクセスすることなく簡単な方法で行われる（詳細後述）。

【００４２】起動された割込ハンドラ31は割込処理を実行し、処理を終了するとＯＳ10内の割込ハンドラ終了処理手段12に割込処理の終了を通知する。割込ハンドラ終了処理手段12は前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる。

【００４３】以上のように、図１の構成においては外部割込が発生したときにＣＰＵ１が直接割込ハンドラ31に処理を移さずにＯＳ10内の割込ハンドラ起動処理手段11に処理を移し、内部レジスタ類の保存と割込要因の解析を行ってから割込ハンドラ31を起動する。

【００４４】このため、割込ハンドラ31はシステムコールによってレジスタの保存などを要求する必要がなくなり、割込処理の内容のみをすべて高級言語で作成することが可能となる。また、ＯＳにおける割込要因の解析が簡単となるため、ＯＳのオーバーヘッドが小さくなる。

【００４５】

【実施例】図２は本発明の実施例のシステム構成図、図３は本発明の実施例のメモリ構成図、図４は本発明の実施例の処理移行状態を説明する図、図５は本発明の実施例のフロー図である。

【００４６】全図を通じ、同一記号は同一対象物を示し、１はマイクロプロセッサ（ＣＰＵ）、1aはプログラムカウンタ（ＰＣ）、1bはスタックポインタ（ＳＰ）、２は割込コントローラ（ＰＩＣ）、3_{-1}，3_{-2}は外部装置、10はＯＳ、11～13はＯＳ10を構成する各部で、11は割込ハンドラ起動処理部、12は割込ハンドラ終了処理部、13は実行管理部である。

【００４７】21，22は二重割込指定部20を構成する部分で、21は割込ベクタ領域、22は二重割込指定テーブルである。また、31は割込ハンドラ、32はアプリケーション・プログラム（ＡＰ）、33はＲＡＭ上に設けらるスタック領域である。

【００４８】図２はＣＰＵ１として８０８６シリーズのマイクロプロセッサを使用した本発明の実施例のマイクロプロセッサ・システムの構成図を示している。説明の便から図２及び図３には２台の外部装置3_{-1}，3_{-2}のみを記載している。以下、図２及び図３により本発明の実施例を説明するが、従来技術と重複する部分については説明を省略する。

【００４９】いま、ＣＰＵ１がＯＳ10の管理のもとでＡＰ32の1200番地を実行中であり、ＰＣ1aが次の1201番地を指している状態で外部装置3_{-1}より割込要求があったものとする。ＰＩＣ２がこれをＣＰＵ１に伝えると、ＣＰＵ１は従来技術におけると同様、ＳＰ1bが指しているスタック領域33にＰＣ1aの指している番地情報"1201"を中断するＡＰ32の再開アドレスとして退避させる。これとともにＳＰ1bはスタック領域33内の次の番地に進み、多重割込に備える。

【００５０】次いで、ＣＰＵ１は従来技術と同様、割込ベクタ領域21のタイプ０の番地にアクセスし、記憶されているアドレス情報"12340" を読み出す。これによってＰＣ1aは 12340番地の次の 12342番地に移るが、前記の 12340番地は従来技術と異なり、割込番号に対応する割込ハンドラ23の先頭番地ではなく、二重割込指定テーブル22の割込番号１に対応する番地である。

【００５１】二重割込指定テーブル22は必要な割込番号の数だけ用意されるが、各割込番号に対応するメモリには同一内容、即ち、ソフト割込の命令が記憶されている。このソフト割込は外部装置よりの割り込みよりも優先度が高いものであるが、この割り込みも外部割込と同様、割込ベクタ領域21にアクセスして処理ルーチンが記憶されているメモリ領域の先頭番地を得るようになっている。ここではソフト割込の割込ベクタ領域21へのアクセスにタイプ２５５（物理アドレスは003FC)を使用することが機械語によって設定されている。

【００５２】上記ソフト割込により二重割込が行われたことになるため、最初の割込でＰＣ1aが移っていたアドレス値"12342" をＳＰ1bが指しているスタック領域33に記憶し、ＳＰ1bは次の番地に移る。

【００５３】ソフト割込が割込ベクタ領域21より得る情報、例えば"20000" は図示のようにＯＳ10K 一部を構成する割込ハンドラ起動処理部11の先頭番地となっている。割込ハンドラ起動処理部11は大別して、割込要因解析部11a とレジスタ保存処理部11b により構成されるが、先のアドレス"20000" は割込要因解析部11a の先頭番地となっており、ここで割込要因の解析が行われる。

【００５４】割込要因解析部11a には割込番号ごとの割込ハンドラの先頭番地が２バイト間隔のアドレスに記憶されているので、割込番号を算出すれば「 20000＋ 2×

（割込番号）－２」の式により割込番号に対応する割込ハンドラのアドレスが判るようになっている。

【００５５】前記割込番号は、スタック領域33に記憶したアドレス値"12342" から割込ベクタ領域21の先頭に記憶されているアドレス値の"12340" を引き、２で割ることにより得られる。即ち、割込要因の解析は簡単な演算を実行するのみで行われ、ＰＩＣ２の内部レジスタ（図示省略）にアクセスするが必要がないため、ＯＳに大きなオーバーヘッドを生ずることがない。

【００５６】この例では上記の演算により“１”が得られ、この割り込みが割込番号１であることが識別され、前記の式より割込ハンドラのアドレスが記憶されている番地が 20000番地であることが求まる。この 20000番地に記憶されているアドレス値を読むことにより割込ハンドラ31の処理に移ることができるが、それに先立ち、レジスタ保存処理部11b がＣＰＵ１の内部レジスタの保存を行う。

【００５７】内部レジスタ類の保存処理は従来技術において割込ハンドラよりのシステムコールによって行う内容と同じであるが、本発明では割込ハンドラ31が関与せずにすべてＯＳ内部のみで行うことができるため、ユーザがレジスタ類の退避に配慮して割込ハンドラを作成する必要がない。なお、このとき、データ・セグメントの付け替えなども行われる（詳細説明は省略）。

【００５８】レジスタ保存処理部11b はレジスタの保存などが終わると割込ハンドラ31を呼び出すが、これは割込要因解析部11a が得た前記 20000番地に記憶されているアドレス値を読むことにより行うが、アドレス値が例えば"3000"であれば、処理を3000番地に移すことにより割込ハンドラ31による処理が開始される。

【００５９】処理が割込ハンドラ31に移されたのちにもし高位の割込要求があったときにはその割込処理に移る。そのため、ＯＳ10は割込ハンドラ31による割込処理中も割込状態即ち、現在の割込処理がどの外部装置よりの要求のものであるかを管理する必要があるが、これは先に割込要因解析部11a により識別された割込番号をＯＳ内部に保存することにより可能である。

【００６０】以上のように、割込ハンドラ31はＯＳによって行われる処理がすべて終了したのちに起動されるため、ＣＰＵ１の内部レジスタを保存したり、セグメントの付け替えのためにアセンブラ言語を使用したり、特別な手順を記述する必要がなく、割込処理内容のみを高級言語だけを用いて記述することができる。その結果、この割込ハンドラ31は通常のサブルーチンと同様に取り扱うことができる。

【００６１】割込処理が終了する場合も割込ハンドラ31は内部レジスタやセグメントに関する特別な記述を行う必要がなく、サブルーチンからメインルーチンに戻す場合に通常使用される命令、例えばＲＥＴＵＲＮ命令でＯＳに処理を戻すことができ、これによりＯＳは割込処理が終了したことを知ることができる。

【００６２】割込ハンドラ31よりの割込終了通知によりＯＳの割込ハンドラ完了処理部12はレジスタの復帰などの処理を行い、中断していたＡＰ32の処理を再開させるが、その処理内容は従来技術とほぼ同一であるので説明を省略する。

【００６３】図４は以上における処理の移行状態を図示したものであるが、図示のように、割込ハンドラは割込処理の際にＣＰＵがレジスタの保存などの処理をすべて終了したのちに呼び出されるため、割込ハンドラ23の処理内容は非常に簡単になる。

【００６４】図５は以上における割込ハンドラ起動時のＣＰＵとＯＳの処理をフロー図に示したものであり、Ｓ１～Ｓ５はフローの各ステップを示す。割込が発生するとＣＰＵのハード処理によって二重割込指定テーブルに無条件で分岐（Ｓ１）させたのち、二重割込指定テーブルにおいてタイプ２５５のソフト割込を発生（Ｓ２）させ、割込ハンドラ起動処理部において割込要因の解析（Ｓ３）とＣＰＵ内部レジスタの保存（Ｓ４）を行う。これらの処理が終わった時点で割込ハンドラを起動する（Ｓ５）。

【００６５】割込ハンドラの処理終了時の細部の処理は前述のように従来技術と若干異なるが、フロー図上では差がないので説明を省略する。以上、図２乃至図５により本発明の実施例を説明したが、図２乃至図５はあくまで本発明の一実施例を示したものに過ぎず、本発明が図示したものに限定されるものでないことは勿論である。

【００６６】例えば、上記の説明は８０８６シリーズのマイクロプロセッサを例として行ったが、本発明がマイクロプロセッサの名称如何に関わらず同様な構成をもつマイクロプロセッサを使用するマイクロプロセッサ・システムに適用可能であることは明らかである。

【００６７】また、図３に示したメモリ構成におけるＲＡＭの記憶内容の一部をＲＯＭに移しても本発明の効果が変わらないことは明らかであり、また、アドレスや記憶内容が図示のものに限定されないことは当然である。更に、ＯＳが保存または付け替え処理を行う対象はレジスタやデータ・セグメントのみに限定されるものではなく、他の名称を有する同種の構成要素を含むことは勿論であり、本発明はマイクロプロセッサ・システムの構成の変形を排除するものではない。

【００６８】

【発明の効果】以上説明したように、本発明においては、ＯＳが内部レジスタ類の保存などの処理をすべて終了したのちに割込ハンドラを起動するため、割込ハンドラにレジスタの退避やデータ・セグメント付け替えのための手順を記述する必要がない。このため、ユーザは外部割込の処理内容を規定する割込ハンドラをすべて高級言語で記述することができ、かつ、記述の簡明化にともない誤処理が発生する可能性が減少する。また、割込要

因の解析が割込コントローラ（ＰＩＣ）にアクセスすることなく簡単な演算によって行われるため、ＯＳのオーバーヘッドが減少する。

【００６９】即ち、本発明は、割込管理を行うマイクロプロセッサ・システムにおける割込ハンドラの作成効率と割込処理の効率の向上に大きく貢献する。

【図面の簡単な説明】

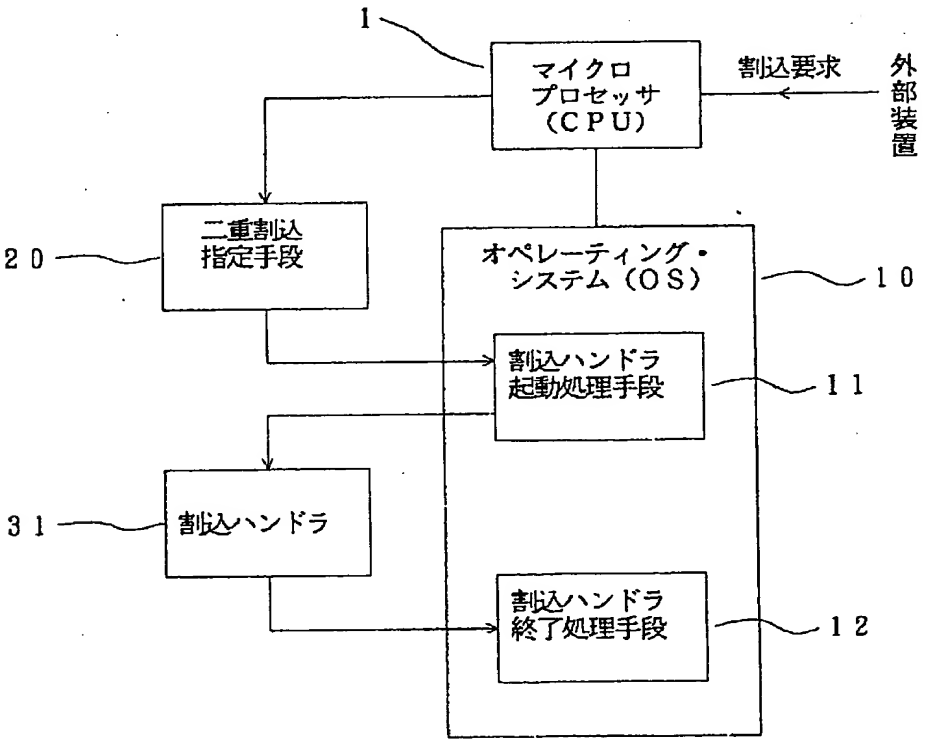
【図１】　本発明の基本構成図
【図２】　本発明の実施例システム構成図
【図３】　本発明の実施例メモリ構成図
【図４】　本発明の実施例処理移行状態説明図
【図５】　本発明の実施例フロー図
【図６】　マイクロプロセッサ・システム構成図
【図７】　従来技術のシステム構成図
【図８】　従来技術のメモリ構成図
【図９】　従来技術の処理移行状態説明図
【図10】　従来技術のフロー図

【符号の説明】

1　マイクロプロセッサ（ＣＰＵ）
1a　プログラムカウンタ（ＰＣ）
1b　スタックポインタ（ＳＰ）
2　割込コントローラ（ＰＩＣ）
3-1～3-8　外部装置（Ｉ／Ｏ）
10　オペレーティング・システム（ＯＳ）
11　割込ハンドラ起動処理手段
12　割込ハンドラ終了処理手段
13　実行管理部
20　二重割込指定手段
21　割込ベクタ領域
22　二重割込指定テーブル
31　割込ハンドラ
32　アプリケーション・プログラム（ＡＰ）
33　スタック領域

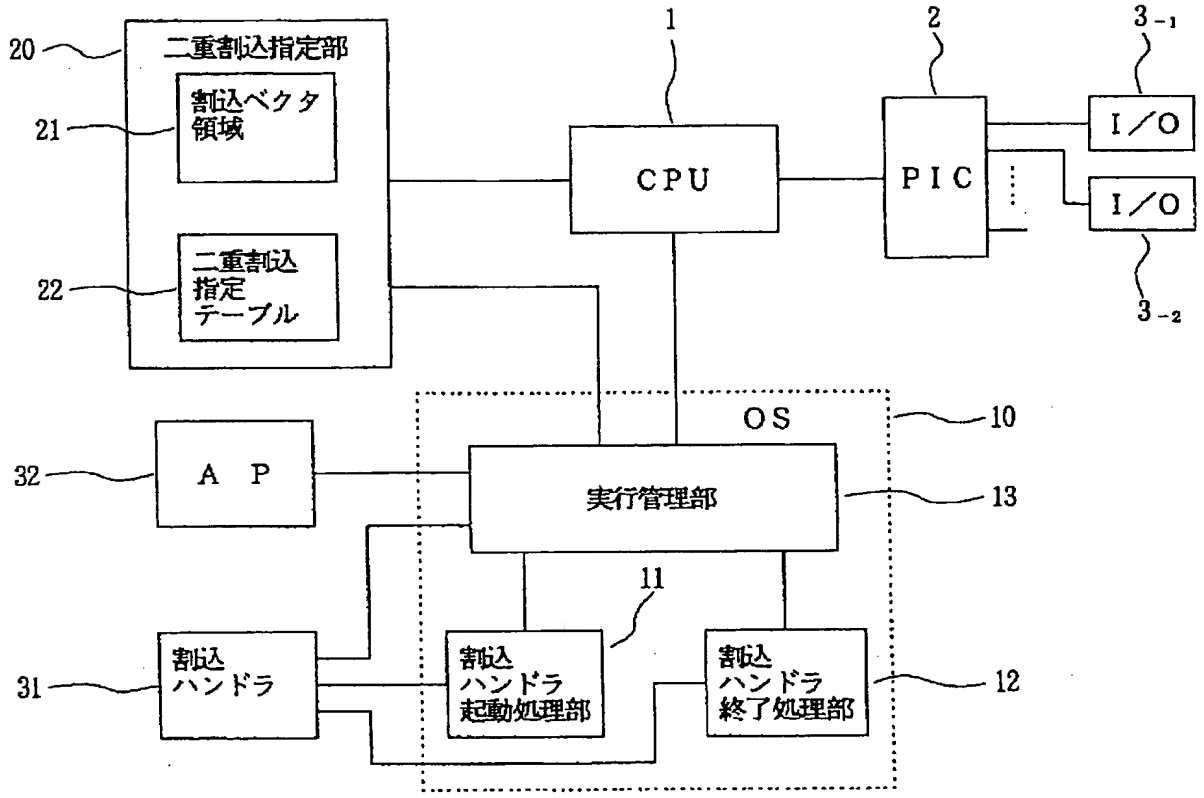
【図１】

本発明の基本構成図

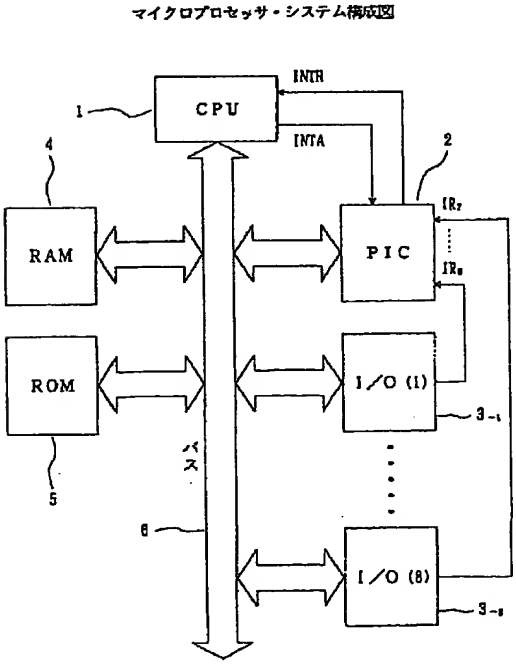


【図２】

本発明の実施例システム構成図

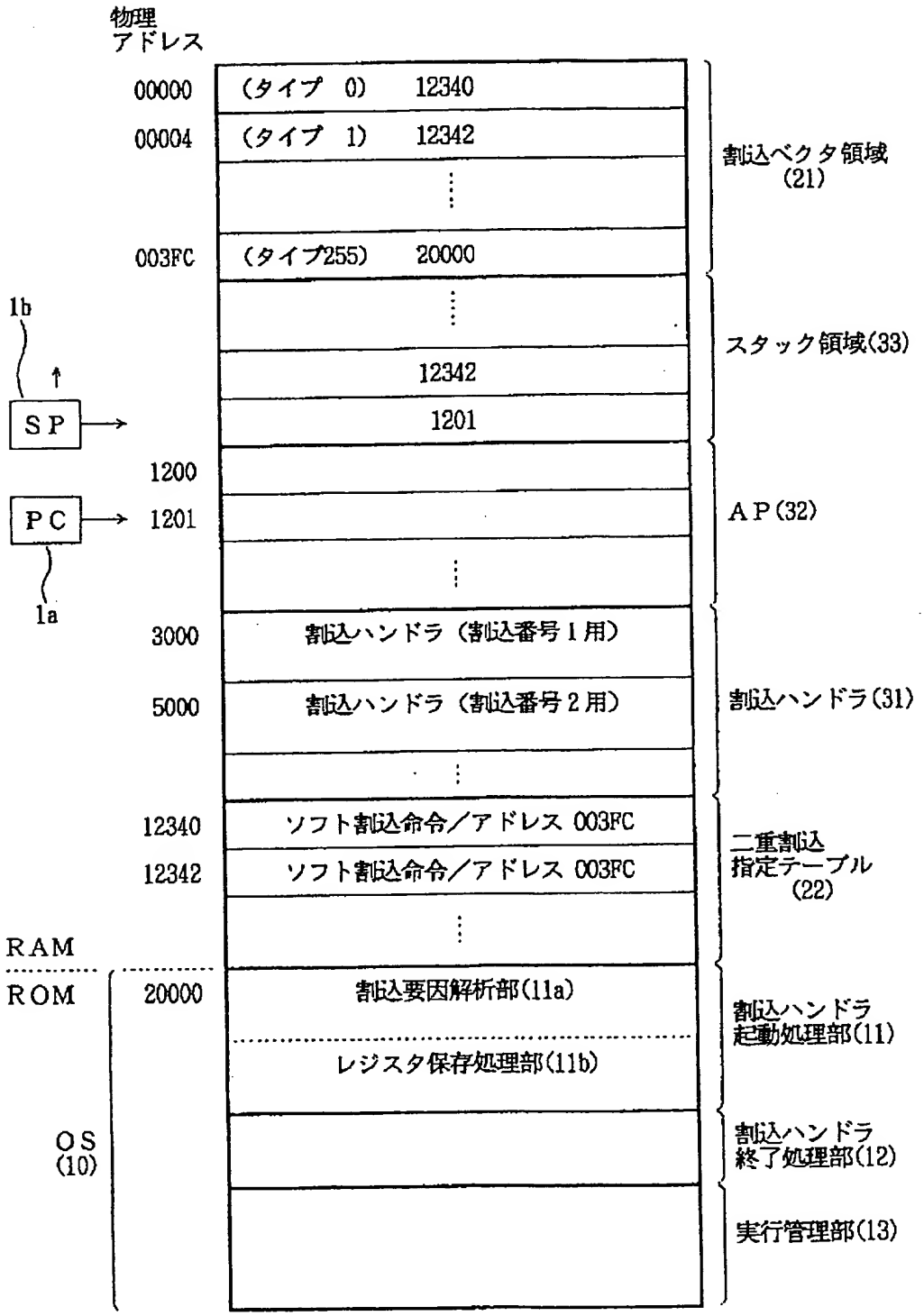


【図６】



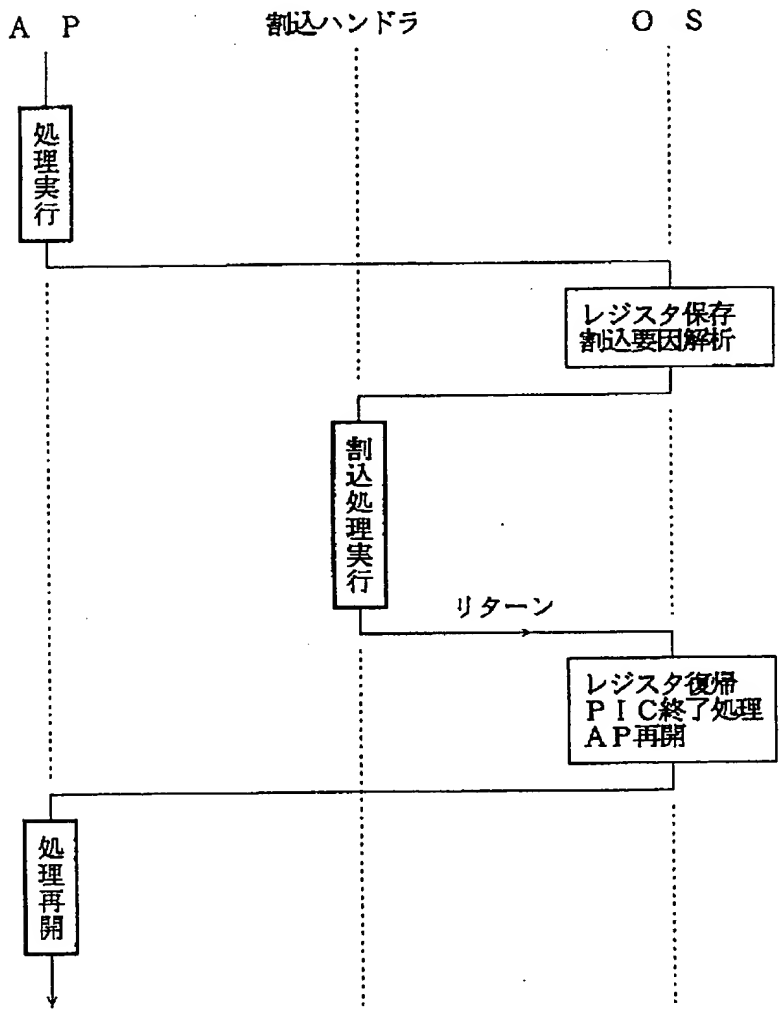
【図３】

本発明の実施例メモリ構成図



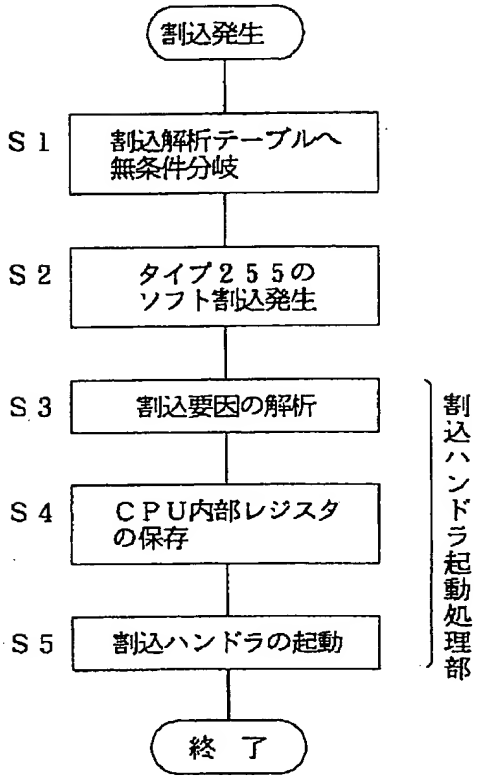
【図４】

本発明の実施例処理移行状態説明図



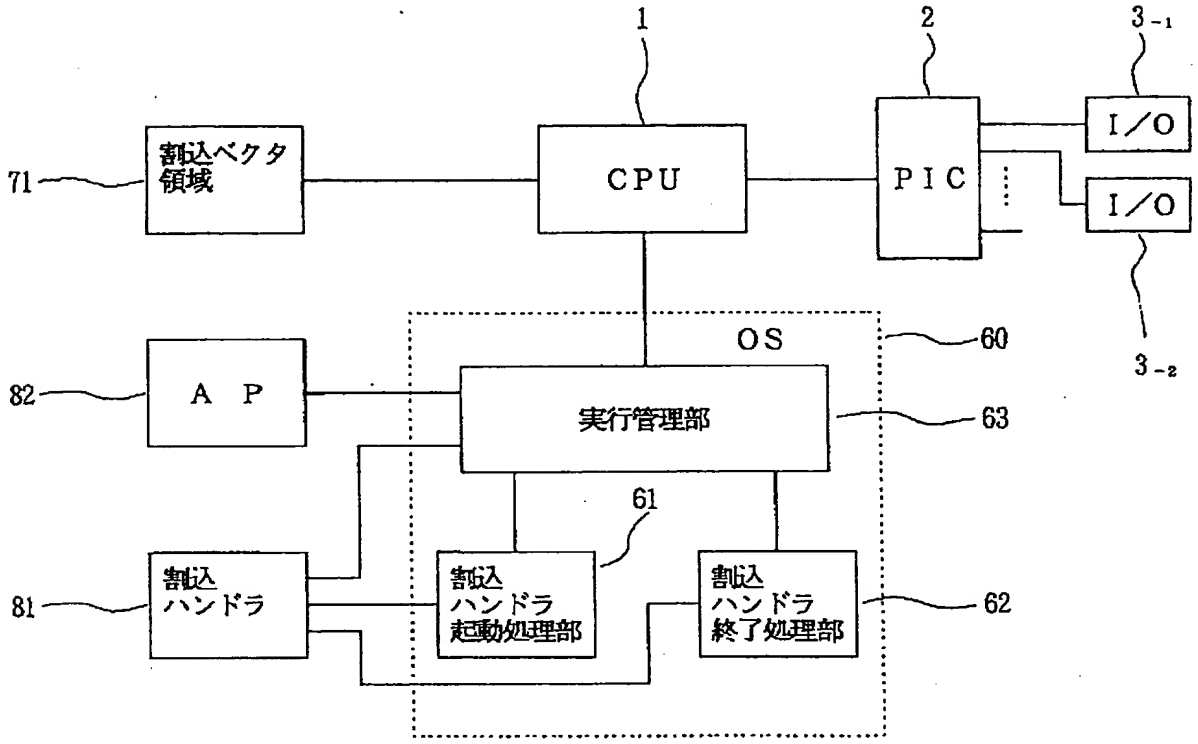
【図５】

本発明の実施例フロー図



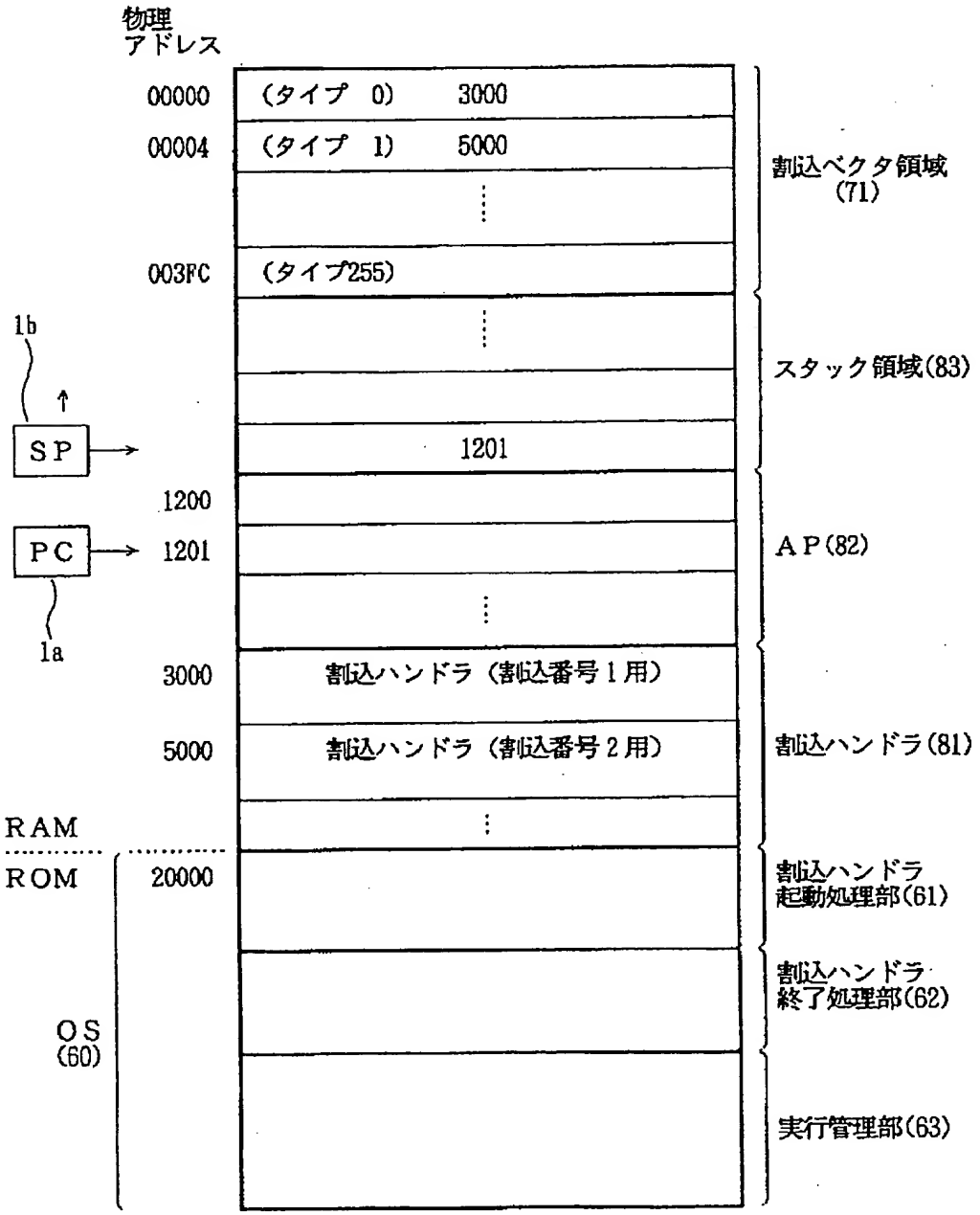
【図7】

従来技術のシステム構成図



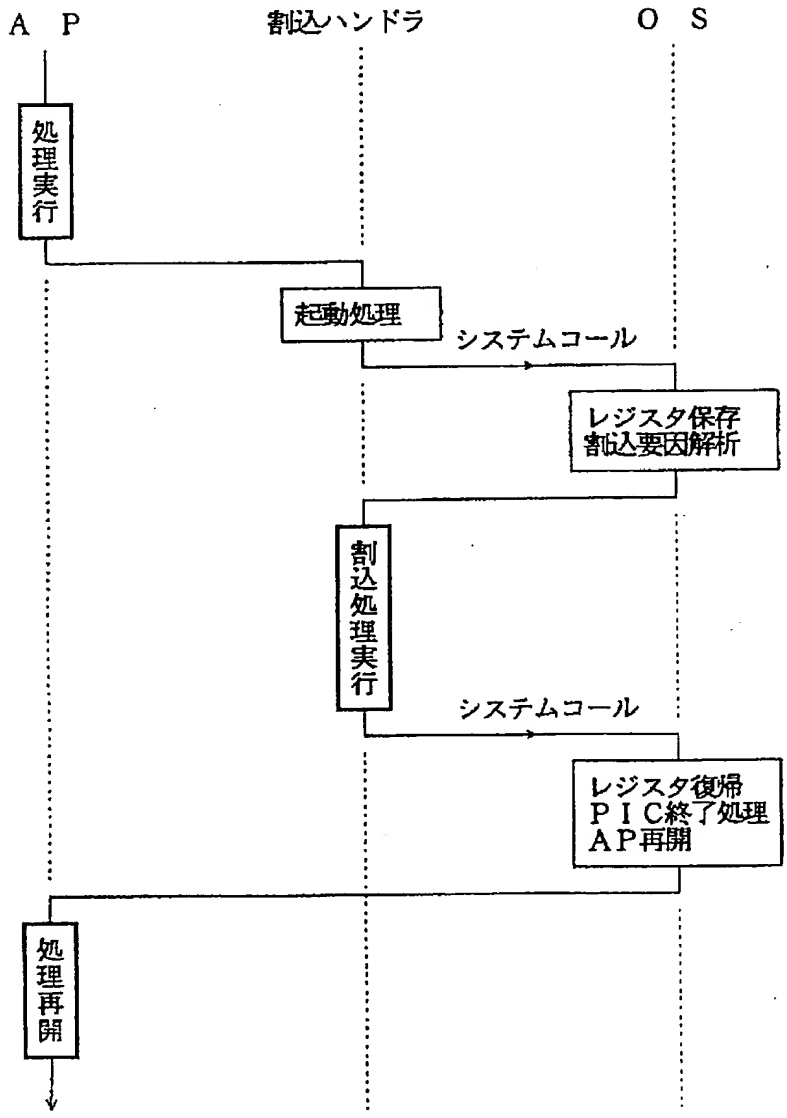
【図８】

従来技術のメモリ構成図



【図９】

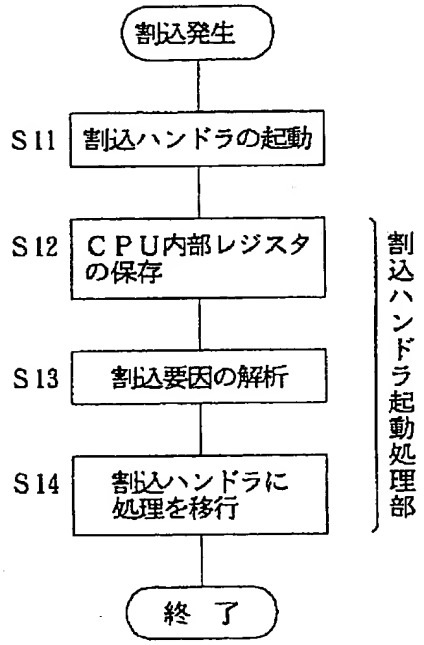
従来技術の処理移行状態説明図



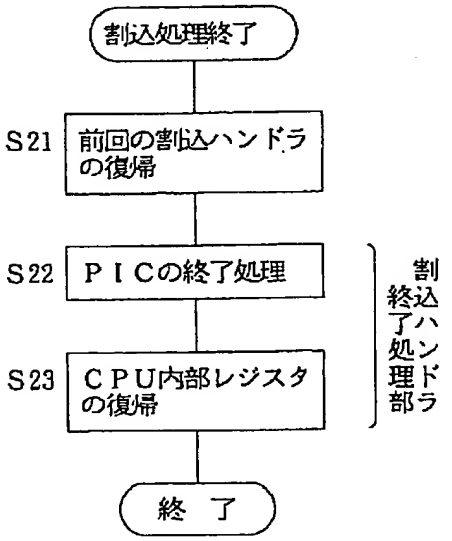
【図１０】

従来技術のフロー図

(1) 割込ハンドラ起動時の処理



(2) 割込ハンドラ終了時の処理



フロントページの続き

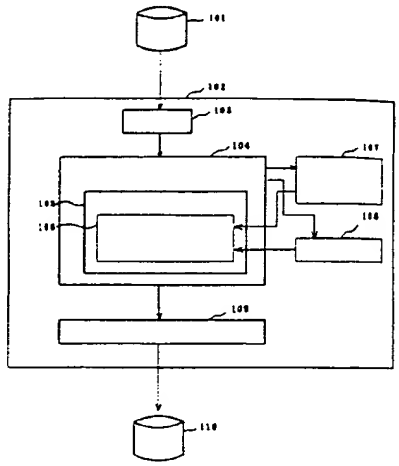
(72)発明者　山脇　智洋
神奈川県横浜市港北区新横浜三丁目９番18号　富士通コミュニケーション・システムズ株式会社内

(72)発明者　熱海　信夫
神奈川県横浜市港北区新横浜三丁目９番18号　富士通コミュニケーション・システムズ株式会社内

(54) LANGUAGE PROCESSING PROGR[redacted]EXECUTING DEVICE
(11) 5-224947 (A) (43) 3.9.1993 (19) JP
(21) Appl. No. 4-22886 (22) 7.2.1992
(71) NEC CORP (72) HIROKO ISOZAKI
(51) Int. Cl[5]. G06F9/45,G06F9/46

PURPOSE: To curtail or delete the size for saving and reset of a internal variable area in pre-and post-processings of an interruption subroutine.

CONSTITUTION: The device is provided with a syntax analyzing part 104 for obtaining interruption subroutine information obtained by analyzing a source program information file containing the interruption subroutine information, and correlation information of a call of each subroutine, and an internal variable arrangement processing part 106 which is contained in the syntax analyzing part 104, and allocates an internal variable of each interruption subroutine and the subroutine called from the interruption subroutine into an exclusive high speed access memory of each interruption suroutine group. Accordingly, there is an effect for improving an execution time of a generated object and shortening the size.

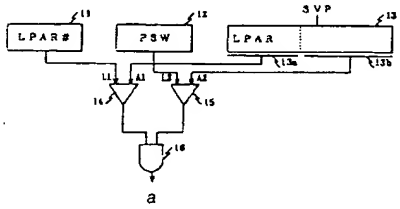


101: source program information file, 102: language processing program executing means, 103: input part, 105: variable arrangement processing part, 107: interruption subroutine information, 108: correlation information, 109: object output part, 110: object file

(54) INTERRUPTION GENERATING CIRCUIT FOR DEBUGGING
(11) 5-224950 (A) (43) 3.9.1993 (19) JP
(21) Appl. No. 4-29201 (22) 17.2.1992
(71) HITACHI LTD(1) (72) KENICHI YAMAMURO(1)
(51) Int. Cl[5]. G06F9/46,G06F11/28

PURPOSE: To efficiently execute a debug by providing a register (LPAR#) for holding a virtual computer number (LPAR number) in an instruction processor, and generating an interruption at the time of coincidence of the LPAR number being in the course of execution and an instruction address.

CONSTITUTION: A comparing circuit 14 compares the contents L1 of an LPAR# 11 and the contents A1 of a compare LPAR number 13a, and outputs "1" or "0" in accordance with a result of this comparison. On the other hand, a comparing circuit 15 compares the contents L2 of an instruction address in a program status word (PSW) 12 and tht contents A2 of a compare instruction address 13b, and outputs "1" or "0" in accordance with a result of this comparison. Subsequently, an AND circuit 16 generates an interruption generating factor signal by AND of the output of the comparing circuit 14 and the output of the comparing circuit 15. This signal is supplied to an SVP 13 an interruption is generated. By storing a value required for the LPAR number 13a and the instruction address 13b, an interruption can be applied by limiting it to only one virtual computer.

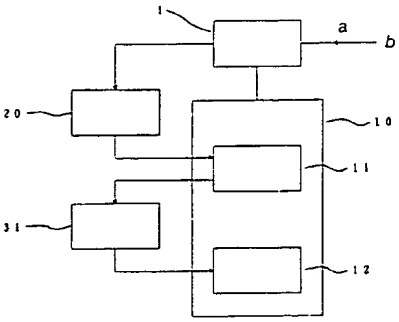


a: interrupting signal

(54) INTERRUPTION MANAGING METHOD FOR MICROPROCESSOR SYSTEM
(11) 5-224951 (A) (43) 3.9.1993 (19) JP
(21) Appl. No. 4-24226 (22) 12.2.1992
(71) FUJITSU COMMUN SYST LTD (72) YASUO NAKAJIMA(2)
(51) Int. Cl[5]. G06F9/46,G06F13/24

PURPOSE: To provide the interruption managing method for generating easily an interruption handler, and decreasing an overhead of an OS.

CONSTITUTION: In the microprocessor system for suspending a processing at the time when a CPU 1 which is executing a processing through an OS 10, receives an interruption request from an external device, and allowing an interruption handler 31 to execute an interruption processing, this system is constituted by providing a double interruption designating means 20 for designating a soft interruption at the time when it is actuated from the CPU which receives the interruption request and migrating the processing to the OS before migrating the processing to the interruption handler, an interruption handler actuation processing means 11 for analyzing an interruption factor and retaining a register, etc., in the OS and actuating the interruption handler, and an interruption handler finish-processing means 12 for executing reset of the register, etc., and an interruption finish processing to the external device in the OS at the time when an end of the interruption processing is informed from the interruption handler, and thereafter, restarting the suspended processing.



a: interruption request, b: external device